웹시스템설계 실습문서 Lab 06

최지헌 unidev@ajou.ac.kr
이재현 dlwogus8888@ajou.ac.kr

2024-10-29

목차

1 학습 목표

- React를 활용하여 웹 어플리케이션을 구축할 수 있다.
- React Component 및 렌더링에 대해 학습한다.

2 ReactJS

이번 실습 문서는 React 공식 홈페이지(https://ko.reactjs.org/)를 참고하여 작성되었다.

React란 사용자 인터페이스를 만들기 위한 자바스크립트 라이브러리이다.

- React는 SPA(Single Page Application) 개발을 위한 Frontend 프레임워크이다.
- React는 선언적이고 효율적이며 유연한 JavaScript 라이브러리이다.
  - 어플리케이션의 각 상태에 대한 간단한 뷰만 설계하면, react가 변경된 데이터에 따라 적절한 컴포넌트만 효율적으로 갱신하고 렌더링한다.
- React는 Component 기반의 설계를 제공함으로써, 구현된 기능의 재사용성을 높이고 이를 통해 생산성이 증가하여 대규모 프로젝트 개발에 유리하다.
  - Component 로직은 JavaScript로 작성되기 때문에 다양한 형식의 데이터를 전달할 수 있고, DOM과는 별개로 상태를 관리할 수 있다.

2.1 React 프로젝트 구성

React 프로젝트를 생성할 때는 npx를 통하여 설치할 수 있다. npx로 create-react-app을 실행하면 된다.

- npx는 npm에서 제공해주는 package 실행 도구이다. Node 14.0.0 혹은 상위 버전 및 npm 5.6 혹은 상위 버전이 필요하다.
- creat-react-app은 React로 웹 어플리케이션을 개발하는 데 필요한 모든 설정이 되어 있는 상태의 프로젝트를 생성해 주는 도구이다.
- React 프로젝트명에는 대문자와 특수기호가 포함되면 안 된다는 점에 주의해야 한다.

아래 명령어를 실행하여 새로운 React 프로젝트를 만들고, 실행할 수 있다.

```
$ npx create-react-app (project name)
$ cd (project-name)
$ npm start
```

react 프로젝트를 생성 및 실행하는 방법

추가적인 정보는 아래의 링크에서 확인할 수 있다.
https://ko.reactjs.org/docs/create-a-new-react-app.html

2.2 React 사용 예시

다음은 생성한 React 프로젝트의 /src/App.js를 아래와 같이 수정하여 React 로고와 지정한 문자열이 보여지는 웹페이지를 localhost의 3000번 port를 통해 보여주는 프론트엔드 예시이다.

```
import logo from "./logo.svg";
import "./App.css";

function App() {
    return (
        <div className="App">
            <header className="App-header">
                <img src={logo} className="App-logo" alt="logo" >
                <p>Hi, React! Toady is 2024/11/14.</p>
            </header>
        </div>
    );
}

export default App;
```

app.js 파일을 수정한 후 프로젝트를 실행한 화면

웹 브라우저를 통해 접속할 경우 위와 같이 표시된다.

이전에 다른 앱을 개발한 적이 있다면, 웹 브라우저를 통한 테스트 시 주의해야 한다. 웹 브라우저가 쿠키나 그 외 기타 다른 캐시 컨텐츠를 남겨 놓았을 경우 테스트가 제대로 진행되지 않을 수 있으니 웹 브라우저를 통해 테스트하고자 할 경우 시크릿 모드를 사용하는 편이 안전하다.

2.3 React Element & Rendering

- 엘리먼트(element)는 React 앱의 가장 작은 단위이며, 화면에 표시할 내용을 기술한다.
- 브라우저 DOM 엘리먼트와 React 엘리먼트의 차이점
  - React 엘리먼트는 DOM 엘리먼트의 가상 표현이다.
  - React 엘리먼트는 일반 객체 (plain object)이며, 쉽게 생성할 수 있다.
    * 여기서 말하는 일반 객체는 컴포넌트 유형과 속성 및 내부의 모든 자식에 대한 정보를 포함하고 있는 일반적인 자바스크립트 객체를 의미한다.

      ```
      <h1 className="headerTag">Hello, World</h1>
      ```

      위 엘리먼트는 실제로 아래와 같은 자바스크립트 객체 형태로 존재한다.

      ```
      {
      ```

```
        type: 'h1',
        props: {
            className: 'headerTag',
            childeren: 'Hello, World'
        }
    }
```

- React 엘리먼트는 불변(immutable)객체이다
  - 엘리먼트를 생성한 이후에는 해당 엘리먼트의 자식이나 속성을 변경할 수 없다.
  - 따라서 React는 변경된 엘리먼트를 화면에 보여주기 위해서, 기존 엘리먼트를 변경하는 것이 아닌 새로운 엘리먼트를 만들어서 기존 엘리먼트를 교체한다.

DOM에 엘리먼트를 렌더링하는 방법

```
//App.js
import ReactDom from "react-dom";

const element = <h1>Hello, World</h1>;
const root = ReactDom.createRoot(document.getElementById("root"));
root.render(element);
```

DOM에 엘리먼트를 렌더링한 결과

2.4 React Component

- React는 컴포넌트를 통해 UI를 재사용 가능한 개별적인 여러 조각으로 나누고, 각 조각을 개별적으로 관리할 수 있다.
- 개념적으로 컴포넌트는 JavaScript 함수와 유사하다. 컴포넌트는 props라고 하는 임의의 입력을 받은 후, 화면에 어떻게 표시되는지를 기술하는 React 엘리먼트를 반환한다.
- 하나의 컴포넌트는 여러 개의 컴포넌트로 구성될 수 있다.
  - 여러 개의 컴포넌트가 모여 하나의 컴포넌트를 구성하고, 이러한 여러 개의 컴포넌트가 또 모여서 또 다른 컴포넌트를 구성하는 형식으로 페이지가 생성된다. 따라서, 개발 중에 수정 사항이 생기면, 그에 맞는 컴포넌트만 수정하면 되기 때문에 개발과 유지 보수 비용이 줄어들 수 있다.
- React 컴포넌트는 함수 컴포넌트와 클래스 컴포넌트로 정의할 수 있다. 현재는 대부분 (hook과 함께) 함수 컴포넌트를 사용한다.
  - 함수 컴포넌트
    - 함수 컴포넌트란, 하나의 props 객체를 받아서 함수에 정의된 React 엘리먼트를 반환하는 함수 형태의 컴포넌트이다. (props와 관련된 내용은 5. React Props에 설명되어 있다.)

```
function Welcome(props){
    return <h1>Hello, {props.name}</h1>;
}
```

- 클래스 컴포넌트
  - 클래스 컴포넌트란, 자바스크립트 ES6의 class 개념을 사용하여 정의한다. React의 모든 클래스 컴포넌트는 React.Component를 상속받는다.

```
class Welcome extends React.Component{
    render(){
        return <h1>Hello, {this.props.name}</h1>
    }
}
```

- 컴포넌트의 이름은 항상 대문자로 시작해야 한다.
  - 소문자로 시작하는 컴포넌트는 , 과 같은 내장 컴포넌트를 의미하기 때문에 그대로 DOM 태그로 인식한다. 따라서 내장 컴포넌트가 아니면서 동시에 소문자로 시작하는 컴포넌트를 사용했다면, 에러가 나거나 원하는 결과나 나오지 않는다.
- React 엘리먼트는 사용자 정의 컴포넌트로도 나타낼 수 있다.

```
function Welcome(){
    return <h1>Hello, World</h1>;
}
const comp = <Welcome >;
```

- 컴포넌트를 렌더링하는 방법 (엘리먼트를 렌더링하는 방법과 동일하다.)

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(comp);
```

2.5 React Props

- 컴포넌트의 입력이자, 컴포넌트의 속성인 props는 컴포넌트에 파라미터로 전달될 다양한 정보를 담고 있는 자바스크립트 객체이다.
- 부모 컴포넌트에서 자식 컴포넌트로 데이터를 넘겨줄 때 사용한다.
- props는 읽기 전용(read-only)이기 때문에 컴포넌트의 자체 props를 수정하면 안 된다.
  - pure 함수란, 입력을 변경하지 않고 항상 동일한 입력에 대해 동일한 결과를 반환하는 함수이다. 아래 함수는 pure 함수의 예시를 보여주고 있다.

```
function sum(a,b) {
    return a+b;
}
```

하지만 아래 함수는 자체 입력을 변경하기 때문에 pure 하지 않다.

```
function withdraw(account, amount) {
    account.total -= amount;
}
```

모든 React 컴포넌트는 props와 관련하여 pure 함수처럼 작동해야 한다. 따라서, 컴포넌트의 자체 props는 수정되어서는 안 된다.

- 컴포넌트에 props 객체를 전달하기 위해서는 컴포넌트를 사용할 때 키-값 형태의 속성을 추가하면 된다.
  - 컴포넌트에서 props 객체를 사용하기 위해서는 props 객체를 전달할 때 사용한 키 값으로 접근하면 된다.

```
function Welcome(props) {
    return <h1>Hello, {props.lastName}{props.firstName}! Are you {props.age}
        old?</h1>;
    // Hello, KimAjou! Are you 22 old?
}

function App() {
    return (
        <div>
            <Welcome firstName="Ajou" lastName="Kim" age={22} >
            // {firstName:"Ajou", lastName="Kim", age:20}인 props를 Welcome 컴
                포넌트에게
                전달
        </div>
    );
}
```

3 실습 과제

3.1 프로젝트 디렉토리 구조

디렉토리 구조는 'create-react-app' 명령어를 통해 생성된 구조를 그대로 제출한다. 단, node_modules 폴더는 삭제한 후에 제출한다. 아래는 실습 수행을 위해 확인해야 하는 파일을 나열하였다.

```
Lab06_(본인학번)/
├── src/
│   ├── App.css
│   ├── App.js
│   ├── index.js
├── package-lock.json
├── package.json
```

3.2 프로그램 구현 내용

웹 페이지에서 가장 기본이 되는 제목과 본문을 React를 사용해서 구현하는 것을 목표로 한다. CSS를 활용하여 중앙 정렬까지 되도록 구현한다. 즉, heading 태그를 활용하여 제목, paragraph 태그를 활용하여 문단을 화면에 표시한다. 아래 그림과 같이, 제목은 "웹시스템설계 10월 29일 (Lab 06)", 문단은 "(본인의 학번) 이름"으로 작성한다.

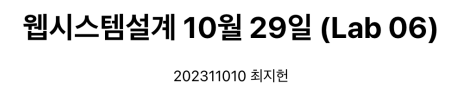

Figure 1: 웹 페이지 화면

3.3 Submission

제출 기한: 10월 29일 자정 / 지각 제출: 10월 31일 자정